Release 19 Guide for APL Programmers

SHARP APL Distributed Sites

I. P. Sharp Associates

SHARP APL/370

APL Systems Development Group
I. P. Sharp Associates Limited
2 First Canadian Place, Suite 1900
Toronto, Ontario, Canada
M5X 1E3

Telephone 416 364-5361
Telex 0622259 I P SHARP TOR

PREFACE

This book describes changes made to SHARP APL by releases 17.1, 17.2, 17.3, 17.4 and 19.0 of SHARP APL/370 for the MVS environment as they pertain to APL programmers.

Related publications. These I.P. Sharp publications are referred to in this book:

SHARP APL/370 Batch Tasks User's Guide (code: 0084 8703 E3)
SHARP APL/370 FILESORT User's Guide (code: 0133 8703 E4)
SHARP APL/370 VSDI User's Guide (code: 0483 8703 E2)
SHARP APL/370 HSPRINT User's Guide (code: 0576 8703 E1)
SHARP APL/370 HCPRINT User's Guide (code: 0347 8703 E3)
SHARP APL/370 IBM3270 User's Guide (IDSH) (code: 0346 8703 E3)
SHARP APL/370 EPH (Emulator Program Handler) User's Guide (code: 0738 8703 E1)
SHARP APL/370 NTOH (Network Terminal Option Handler) User's Guide (code: 0732 8703 E1)
SHARP APL/370 TSIO User's Guide (code: 0461 8703 E19)
SHARP APL Reference Manual (code: 79RM05)

Sites which are running any of the above products will have received their respective user guides. To get more copies of any of these publications, please contact your I.P. Sharp representative.

Printed in Canada
March 1987
Publication Code: 0375 8703 E19

CONTENTS

INTRODUCTION, 1

LANGUAGE ENHANCEMENTS, 1
- Table Function, 1
- Re-defining Names on the *SI* stack, 2
- Enhancements to ↑ and ↓, 4
- Fill Elements for Enclosed Arrays, 4
- Enhancements to □*FMT*, 5

ENHANCEMENTS TO FACILITATE APPLICATION DEVELOPMENT, 5
- Constrained User Support, 5
- □*TRAP OFF* Action, 6
- Batch Task Scheduler, 7
- Miscelleous Enhancements, 7

□*SC* EXTENSIONS AND CHANGES, 8

)*CC* NON-APL ASYNCHRONOUS TERMINAL SUPPORT, 9

□*SJ* SYSTEM JOURNAL FACILITY, 10

PERFORMANCE IMPROVEMENTS, 11
- APL Primitives, 11
- Direct Support for Rank Operator, 11
- File System, 12

SYSTEM SUPPORT SOFTWARE, 12
- FILESORT, 12
- HSPRINT, 13
- HCPRINT, 13

MULTIPLE LANGUAGE SUPPORT, 15

AUXILIARY PROCESSORS, 15
- DB2 Interface (AP127), 15
- TSIO, 16
- AVAM, 16
- VSDI, 17
- FCAP, 17
- IDSH, 17
- Full Screen Manager (AP124), 18
- S-task Interface (AP1), 19
- EPH -- Support for Asynchronous Terminals, 19

MISCELLANEOUS, 21
- ISO Conformance, 21
- Improved CPU Accounting and Timing Precision, 21
- Improvements to Mathematical Functions, 21
- Minor Changes and Corrections, 21
- Communications, 22
- Obsolescent and Obsolete Facilities, 22

APPENDIX A: Changes to Internal Form of APL Sentences, 25

APPENDIX B: Reduction Operator

INTRODUCTION

This Guide describes salient features of SHARP APL Release 19 for the APL programmer. Highlights:

- New "table" function ⍪⍵
- Functions on *SI* stack may be redefined
- Abbreviated left arguments to ↑ and ↓
- Fill element for enclosed arrays
- ⎕*FMT* accepts enclosed right arguments
- Constrained user support for closed applications
- Time-outs for shared variable waits through ⎕*SC*
- New DB2 interface
-)*CC* improved support for non-APL asynchronous terminals
- ⎕*SJ* system journal facility
- Performance improvements in space and time
- Improved CPU accounting precision

LANGUAGE ENHANCEMENTS

Language enhancements provide more convenient and direct ways for programmers to achieve given results, thus improving productivity. There are several such enhancements in this Release. Besides offering improved functionality to APL programmers, these enhancements offer performance improvements over previous APL expressions to achieve the same computation.

In addition to the enhancements mentioned in this section, ⎕*SC* has been extended in a significant way. It is detailed later.

Table Function

The "table" function (⍪⍵) has been implemented as described in K.E. Iverson's "A Dictionary of APL" (publication code: 0402 8412 E1). The character ⍪ "comma-bar" is formed by overstriking , "comma" and - "bar".

⍪⍵ might be thought of as a variation of ravel (,⍵). Whereas ,⍵ returns a rank-1 (vector) result, ⍪⍵ always returns a rank-2 (matrix) result (hence the name "table").

⍪⍵ is defined as:

```
      ⍪⍵ ↔ ((1↑(⍴⍵),1),×/1↓⍴⍵)⍴⍵
```

,⍵ by comparison, is defined as:

```
      ,⍵ ↔ ((0↑(⍴⍵),1),×/0↓⍴⍵)⍴⍵
```

Examples:

```
      ⍪0 1 2
0
1
2
      ⎕NAMES⊃⍪⎕NUMS
   1854339 APL          1
   1505970 APEWORKFILE 2

      ⍴⍪X←2 3 4 5⍴⍳999
2 60
      ⍴⍪⍤3 X
2 3 20
      ⍴⍪⍤2 X
2 3 4 5
      ⍴⍪⍤1 X
2 3 4 5 1
      ⍪¯1⍤<' THESE ARE THE WORDS'
THESE
ARE
THE
WORDS
```

Re-defining Names on the *SI* Stack

⎕*FX* and 3 ⎕*FD* have been extended to apply to all functions, regardless of the presence of the resulting function name on the *SI* stack. The new definition of the function is established and the *SI* stack adjusted to reflect the new definition.

The primary purpose of this extension is to allow APL debugging tools and function editors, such as the *DEL* function in workspace 7 *DEL*, to operate on pendent or suspended functions. Other uses include the writing of self-modifying code, such as adding ⎕*PNAMES* of a package to the header of a function before doing the ⎕*PDEF*.

Terms:

Active Function
: The currently executing function. This function is the one that executed the ⎕*FX* or 3 ⎕*FD*.

Pendent Function
: A function that has partially completed execution of a line of code and has called another function, ⍎, or ⎕. Such functions appear in)*SI* without a *.

Suspended Function
A function that has been interrupted by an error, event, or *SΔ* stop. Such functions appear in *)SI* with a *.

Definition of Extension:

Deleting Local Names
When a local, label, or argument is deleted from the function definition, the next more global value is exposed. Shares for deleted names are retracted.

Inserting Local Names
New locals added to the header initially have the same default value they would have been given at function entry. For example, a new local *□TRAP* would be assigned an empty vector. Most variables receive no value. New labels receive their line numbers as their values.

Moving Local Names
When a local is moved from one line to another, the local is first deleted from its old position and then inserted in its new position. For example, if *L*1 is a shared variable and is moved to become the label on line 5, the share will be retracted and the value 5 will be assigned to *L*1. If a local is moved to the result (in line 0), that name will then become the result of the function. Any names defined in line 0 may be rearranged without altering their values. Labels moved to line 0 become undefined variables.

Changing Active and Pendent Function Lines
Partially executed lines in active and pendent functions cannot resume unless the new definition of the line is identical to the old one. The comparison is done using the internal representation of the old and new lines. For most functions, this is equivalent to comparing the display of the lines, but some lines of very old functions may differ internally even though their displays are the same. All lines of locked functions are treated as different from the new definition. Inability to resume a changed line is reported as *SI ERROR* (error number 9). Suspended functions may be restarted by *→□LC*.

Changing the Value or Nameclass of a More Local Function
Inserting or deleting the name of a function invoked more locally than the changed function causes the local function to be deleted from the *SI* stack. Blanks appear for function name and line number in the *SI* display. An *SI ERROR* will result from an attempt to resume such a function. Any branch will exit from the function. *□EC* and *□TRAP* defined in such a non-restartable *SI* stack entry will still be honoured. Note that inserting a recursive function's name in its header will cause all but the most global invocation of the function to be non-restartable.

Finally, in addition to this improved functionality, *□FX* and 3 *□FD* now run up to 1.5 times faster than in Release 17. For more details, refer to Appendix A.

Enhancements to ↑ and ↓

The left argument to ↑ and ↓ may now be abbreviated -- a short left argument is padded with 0's in the case of ↓, or with a trailing segment of the shape of the right argument, in the case of ↑. For example:

```
      □←M←3 4⍴⍳12
1  2  3  4
5  6  7  8
9 10 11 12

      ¯2↑M
5  6  7  8
9 10 11 12

      2↓M
9 10 11 12
```

Note the added utility of this extension in conjunction with the rank operator.

```
      X←2 3 4 5⍴5
      ⍴6↑X
6 3 4 5
      ⍴6↑⍤3 X
2 6 4 5
      ⍴6↑⍤2 X
2 3 6 5
      ⍴6↑⍤1 X
2 3 4 6
      ⍴6↑⍤0 X
2 3 4 5 6
```

As well, ↑ and ↓ now incorporate direct support for the rank operator ⍤, so that instances of ↑ and ↓ with the rank operator (α↑⍤R ω and α↓⍤R ω) execute significantly faster, with less chance of getting *WS FULL*. Speed-up factors are typically from 3-10 in cases involving the rank operator. Other cases are 10% slower to 30% faster.

Fill Elements for Enclosed Arrays

<⍳0 has been defined as the fill element for overtaking or expanding enclosed arrays. For example:

```
      □PS←¯1 ¯1 ¯2 ¯3
      1 0 1 \ 'GREAT'⊃'STUFF'
┌─────┐ ┌┐ ┌─────┐
│GREAT│ ││ │STUFF│
└─────┘ └┘ └─────┘

      7↑'THE'⊃'QUICK'⊃'BROWN'⊃'FOX'
┌───┐ ┌─────┐ ┌─────┐ ┌───┐ ┌┐ ┌┐ ┌┐
│THE│ │QUICK│ │BROWN│ │FOX│ ││ ││ ││
└───┘ └─────┘ └─────┘ └───┘ └┘ └┘ └┘
```

Releases of SHARP APL prior to Release 19 signalled *NONCE ERROR* in such cases. Applications currently depending on the nonce error to determine storage bype should be examined.

Enhancements to ⎕*FMT*

⎕*FMT* has been extended to accept a right argument which is a vector of enclosed arrays. The expression

```
LHS ⎕FMT A⊃B⊃C
```

is equivalent to the expression

```
LHS ⎕FMT (A;B;C)
```

Example:

```
      A←⍳4 ⋄ B←.5 ⋄ C←'ABCD'
      'I1,F5.1,A1' ⎕FMT A⊃B⊃C
1  0.5A
2     B
3     C
4     D
```

The extension allows the right argument to be computed without awkward and inefficient expressions involving execute (⍎⍵). This facilitates generation of customized reports in 4th generation applications.

ENHANCEMENTS TO FACILITATE APPLICATION DEVELOPMENT

Constrained User Support

Release 19 of SHARP APL contains enhanced support for closed applications. There are various types of closed applications, such as:

- Applications that the user need not know are embedded in an APL system. A securities-clearing system used by non-programmers is an example of such a system.

- Applications where the designer, for reasons of security or ergonomics, wants to ensure that the user cannot exit into "normal" APL. That is, the application desires to retain continuous control of the session.

Closed applications are typically menu-driven, and may be used by people who know little or nothing about APL. Such users tend to be ignorant of the syntax of APL expressions and commands, and get frustrated when they end up in immediate-execution mode as the result of a hardware or software error. In addition, the *CONTINUE* workspace resulting from intentional or accidental disconnections may leave the user in limbo when an attempt is made to establish a new session.

Release 19 introduces the concept of a "constrained user". The constraint is associated with a user number. It is intended to

reduce the work that an application programmer must do to establish an environment which isolates the application user from APL. The behavioral differences of a constrained account from ordinary accounts are as follows:

- At task initiation, the *PROFILE* workspace is always loaded, even if a "hot continue" workspace exists, and would have been loaded for an unconstrained user. The application may, of course, ⎕*LOAD* the *CONTINUE* workspace, if desired.

- Return to immediate execution will force a)*OFF* for the task, even if an extant 2001 ⎕*TRAP*-event would trap the return on an unconstrained account.

- A *SWAP ERROR* (hardware error) or *SYSTEM ERROR* (software error) will force a)*OFF* for the task.

- ⎕*RUN* from one user number onto a different number is prohibited, if the task being initiated would be running on a constrained user number. The intent of this prohibition is to prevent overwriting of the *PROFILE* workspace on the constrained user number.

- For each active APL task, an ATASK bit (ATIFLCON) indicates whether or not the task was a constrained account when the task was initiated. This information, together with facilities provided for control of output display in SITE exits, allows the task to discard or modify the)*HI* text issued at task initiation, as well as the other text normally issued at task initiation and termination.

- For all S-tasks and T-tasks, (2 ⎕*WS* 3)[⎕*IO*+3] has the value 1 if there was a "hot continue" workspace present at task initiation, and a 0 otherwise. N-tasks and B-tasks always observe a value of 0. The value reflects only the state at task initiation, and does not reflect later state changes, such as:
  - the continue workspace being loaded by another task
  - the continue workspace being)*DROP*'ed
  - the continue workspace being overwritten by another task on the same user number, terminating abnormally.

⎕*TRAP OFF* Action

Specification of *OFF* as a keyword in the *D* action code will cause the task to be signed off when the task returns to immediate execution:

```
⎕TRAP←'∇ 2001 D OFF'
```

The *OFF* keyword effectively issues)*OFF* for the task when this trap event is taken. This differs from ⎕*BOUNCE*, which effectively issues)*CONTINUE HOLD*, and also differs from *CLEAROUT ⋄ WALKWAY*, which effectively issues)*OFF HOLD*.

The *OFF* keyword is expected to be of greatest benefit to closed applications, in which a continue workspace is often considered to be a liability rather than an advantage.

Batch Task Scheduler

The Batch Task Scheduler has been enhanced in the following ways:

NTASK(RETRY)

A new optional parameter has been made available for use with the *NTASK* request processing specification.

When the Batch Task Scheduler attempts to start a scheduled N-task, initiation may fail due to the system being full or because of the user task limit. If (*RETRY*) is not specified in the request, then the Scheduler will mark the request as *FAILED*. If (*RETRY*) is specified, the request will be left in the *QUEUED* state and the Scheduler will make further attempts to start the N-task.

ABNORMRST([*LOAD*][*CONTINUE*])
NORMRST([*LOAD*][*CONTINUE*])

Optional sub-parameters may now be specified for *ABNORMRST* and *NORMRST*, to control request restart. The Batch Task Scheduler attempts to restart requests that were active at the time of system shutdown or crash, if the request is marked *NORMRST* (system shutdown), or *ABNORMRST* (system crash) respectively.

In previous releases, requests were always restarted using the *CONTINUE* workspace, as specified in the □*RUN* parameters. This behaviour is now also observed when (*CONTINUE*) is specified as a sub-parameter, or if no sub-parameter is specified. When (*LOAD*) is specified, request restart uses the load workspace.

See the "Batch Tasks User Guide" (publication code: 0084 8703 E3) for more information.

Miscellaneous Changes

PROFILE Security

PROFILE workspaces are now protected in the same way that *CONTINUE* workspaces are: the workspace cannot be loaded from another user number, unless the proper signon lock for the workspace owner number is specified.

Storage for *PROFILE* workspaces are no longer billed.

During on-line FINCDUMP, only the volume class actually being FINCDUMPed is interlocked.

If your system is configured to use a Master Directory, this means that users will not be prevented from tying files, in one volume class, during the time that reorganization is being done on another volume class.

□*SJ* System Journal Facility

□*SJ* can be used to collect information about APL application components. □*SJ* is described later in this guide.

□*SC* EXTENSIONS AND CHANGES

□*SC* now allows a □*SC* wait to terminate after a user-specified time interval. This allows an application to wait on □*SC* without a possible indefinite hang if a partner fails to respond.

□*SC* may be **set** to any scalar real number *ω*; a **reference** to □*SC* will complete within *ω* seconds after the time of the set. For example,

```
□SC←15
   (10 seconds elapsed time of real processing)

□SC
   (5 second delay - assuming no shared variable posts)
```

Invalid settings to □*SC* are ignored. A negative or zero value is legal, and will result in no delay.

A set of □*SC* (□*SC*←*K*) will no longer purge an outstanding shared variable post.

If □*SC* is validly set for a timeout, and a reference to □*SC* terminates because of a shared variable post, the timeout remains pending. For example,

```
□SC←15

□SC
   (terminates before 15 seconds after set, because of shared
   variable post)

□SC
   (terminates 15 seconds after set, if no more shared variable
   posts occur)
```

If a reference to □*SC* completes because of timeout, another reference to □*SC*, assuming no intermediate set of □*SC*, will not terminate immediately, but will suspend indefinitely until a shared variable post occurs, or until the □*SC* wait is broken by a weak interrupt.

The result of □*SC* is compatible with previous releases of SHARP APL. In particular, the result of □*SC* will always be scalar 1, unless the APL task cannot sign on to the SVP because its clone ID is in use. In such cases, □*SC* will terminate immediately with a result of 0.

Treatment of breaking out of a □*SC* wait has been changed to allow it to complete on receipt of a weak interrupt (one break), after which execution continues. Previously, breaking out of a □*SC* wait required receipt of a strong interrupt (two breaks).

)*CC* NON-APL ASYNCHRONOUS TERMINAL SUPPORT

The new system command)*CC* (character control) provides enhanced support for non-APL asynchronous terminals. The syntax of)*CC* is:

```
)CC [OLD|NEW]
```

The default value is *OLD*; an empty argument returns the current setting. Successful setting of)*CC* returns the previous setting.

)*CC* can be entered before sign-on in either the first or the second alphabet. (However, in such cases, the old setting is not displayed, and an argument must be supplied.)

When)*CC NEW* is entered from a non-APL asynchronous terminal:

- (2 ⎕*WS* 3)[⎕*IO*+4] is 1.
- (2 ⎕*WS* 3)[⎕*IO*+11] changes from 3 to 9.
- First alphabet is lower case (input and output).
  Second alphabet is upper case (input and output).
  Third alphabet is lower case (output only).
- BACKSPACE is "destructive"; that is, characters backspaced over are assumed to be erased. However, "o BACKSPACE u BACKSPACE t" still results in *INPUT INTERRUPT* (and not in the character "t").
- Graphic mappings have been rationalized, so that input characters are mapped to the restricted set of characters which resemble the graphic on the device. In particular, there is no mapping for characters such as quad (⎕) or divide (÷).

The assignment of such characters is as follows (⎕*AV* positions assume 0 origin):

⎕*AV* Position	Printable Character
4	$ Dollar
14	[Left bracket
15	] Right bracket
16	(Left parenthesis
17	) Right parenthesis
18	; Semi-colon
19	/ Slash
20	\ Backslash
26	+ Plus
27	- Minus
30	* Star (power)
34	^ Caret (circumflex)
36	< Less than
38	= Equals
40	> Greater than
47	, Comma
48	! Exclamation mark
53	? Query
54	~ Tilde
61	_ Underscore

82	}	Right brace
83	{	Left brace
150	.	Period
152		Blank
153	'	Quote
154	:	Colon
197		Broken stile
198	`	Grave accent
199	#	Octothorpe
200	@	At sign
201	"	Double quote
206	%	Percent
207	&	Ampersand

When *)CC NEW* is entered from an APL terminal:

- (2⎕*WS* 3)[⎕*IO*+4] is 1.
- no other change

When *)CC OLD* is entered from either an APL or non-APL terminal:

- (2⎕*WS* 3)[⎕*IO*+4] is 0.
- (2⎕*WS* 3)[⎕*IO*+11] changes from 9 to 3 if terminal is non-APL.
- Character set mapping reverts to the old definition if previous state was *NEW*.

⎕*SJ* SYSTEM JOURNAL FACILITY

The SHARP APL environment provides new facilities that enable a site to collect management information about the APL system and application components. The new SJAP auxiliary processor, as well as the APL shared system variable ⎕*SJ*, are used in instrumenting APL applications and system components.

Management information can be made available in the areas of resource and facilities utilization, and can be used for:

- Calculating charges for use of resources
- Surveying usage of system and application software
- Planning for long-term capacity requirements

Start-up parameters allow a site to control which parts of the APL system are to be instrumented. Site exits may be used to control and direct, the format and destination of logged data (e.g., to SMF).

IMPROVED PERFORMANCE

The performance of the Sharp APL system has been improved in several ways:

The APL internal SVC rate has been reduced, by about 65% on a typical heavily-loaded system. This will reduce CPU time associated with immediate execution transactions and file operations.

A more efficient algorithm is now used by the swap scheduler. Large sites may observe a reduction in the CPU time spent scheduling tasks of 40% or more.

As well as these general improvements, performance gains have been made in the following specific areas:

APL Primitives

- The definition of a "temp" has been extended to include references whose reference count is exactly one.

  The benefits are reduced chances of *WS FULL* and faster execution by ravel (*,ω*), scan (*F\ω*), *⎕PVAL*, and *⎕PLOCK*, when the right argument has a reference count of one.

- *⎕FI*, *⎕VI*, *⍎*, *⎕FX*, and 3 *⎕FD* now run faster. *⎕FI* and *⎕VI* now run approximately 2.5 times faster; *⎕FX* and 3 *⎕FD* run up to 1.5 times faster.

- The reduction operator (*F/ω*) has been completely rewritten, to give improved performance, and to provide conformance with the ISO standard. See Appendix B for more information.

  The performance of reduction is now generally equal to or better than that in Release 17, except along short axes. Improvements vary considerably depending on the function and characteristics of the data, but can be up to 70%.

- The availability of the table function (*⍪ω*) can be exploited to give improved performance.

  For instance, the VIEWPOINT report writer module saw a modest performance gain with the use of the table function.

- *⎕PACK* and *⎕PDEF* run up to 10% faster.
- Matrix divide and determinant functions run up to 35% faster.

Direct Support for Rank Operator

Dyadic ↑ and ↓ and monadic , and ⍪ incorporate direct support for the rank operator:

```
α↑⍤R ω
α↓⍤R ω
,⍤R ω
⍪⍤R ω
```

The benefits are twofold: one, the chances of *WS FULL* are reduced; and two, the derived functions execute considerably faster. ↑⍤*R* and ↓⍤*R* are faster by factors of 3 to 10, and ,⍤*R* by 2 to 500. (The speed-up factors depend on the data.)

File System

⎕*LIB* no longer forces workspace compaction. As a result of removing the mandatory workspace compaction for ⎕*LIB*, it now runs faster most of the time.

⎕*NAMES* now runs approximately 7 times faster.

Improved File System - APL Communications

The internal SVC rate for file functions has been reduced by up to 75% (as measured on a typical load on the I.P. Sharp internal system).

File operation CPU time has been reduced by up to 15%.

A flag has been added to the file system dataset tables indicating when a given dataset is out of free records.

This improvement eliminates the need for the file system to search through a dataset in order to discover that it is full. Performance improvement is realized particularly when the file system is running short on free space.

During on-line FINCDUMP, only the volume class actually being FINCDUMPed is interlocked.

For systems configured to use a Master directory, this means that users will not be prevented from tying files, in one volume class, during the time that reorganization is being done on another volume class.

SYSTEM SUPPORT SOFTWARE

FILESORT

Numeric data types

FILESORT is no longer sensitive to the internal representation of numeric data. Numeric data types may be mixed. The resulting file will have the least data type, of fixed point and floating point, required to express the data, regardless of the data types of the input. Data of complex internal data type is permitted but must have an imaginary part of zero, so that it may be converted to fixed or floating point.

Restrictions on Keys

Prior to Release 19, FILESORT imposed a restriction of 12 keys with a total key length of 256. Release 19 extends maximum allowable total key length to 4088 and a maximum number of keys to 20.

Enclosed Arrays

Files of mixed data type (numeric and character) may be sorted. Input and output components must be enclosed vectors of numeric and character matrices. I.P. Sharp's VIEWPOINT system exploits this new feature, to result in a 20% decrease in the cost of sorting mixed data type files.

For more information, see the SHARP APL/370 FILESORT User's Guide (publication code: 0133 8703 E4).

FILESORT Use of System Files

The queue file's name is now specified by a start-up parameter.

The interlock file's name is the same as the queue file's name with the first letter underscored.

The SORTBILL file now has 11 elements per entry, versus 7. The additional 4 elements are the encoded queue file name.

HSPRINT

HSPRINT's submission function, *HSP*, now accepts multiple destinations after the *DEST*= sub-parameter in its print specifications. For example, the *DEST*= sub-parameter can be of the form:

```
PRINT(DEST=(DEST1,DEST2,DEST3,...,DEST128))
```

Previously, HSPRINT output could be routed to one user-specified JES node only.

To determine the list of valid destinations supported by the I.P. Sharp Timesharing System, use the interactive mode of *HSP* (`0 HSP ''`), and enter *HELP* in response to the query *PRINT SPECS*:.

HCPRINT

Two new user functions, *HCDEFNS* and *HCNAMES*, are now included in workspace 1 *HCPRINT*. For more information, see the SHARP APL/370 HCPRINT User's Guide (publication code: 0347 8703 E3).

R←HCNAMES

Returns a matrix of all the APL node names that may be used as print destinations in the right argument to the *HCSUBMIT* function or input to *HCREQ*.

R←HCDEFNS 'namelist'

Returns information for APL node names specified in the right argument **namelist**. The result of *HCNAMES* could be used as the argument to *HCDEFNS* to obtain information for all APL node names. The result contains the APL node name; the user numbers to which use of the node is restricted (0 indicates unrestricted access); the node description; and the list of VTAM LU names that constitute the print node.

New NOBANNER request option

This new HCPRINT request option allows authorized users to avoid printing of banner pages. This is of great utility when expensive, pre-printed forms are being used.

Improved Device Support

Four new HCPRINT request options, DEPTH, WIDTH, LPI, and CPI have been added to enhance device support. This enables users to more effectively utilize device features. The effect of these options differs according to the devices' abilities to process configuration commands.

Devices which are "SCS-Enquirable" are able to process these commands. For such devices, HCPRINT will set the devices' configuration for DEPTH, WIDTH, LPI, and CPI, if valid, and format the output to suit this configuration.

For all other devices, HCPRINT assumes that the user has configured the device as specified in the options, and will format out to suit.

HCPRINT Performance Improvements

Improvements have been made to the request processing logic to reduce CPU consumption.

Support for SCS devices has been improved by sending all the output for one HCPRINT request as a single VTAM chain. This reduces the interaction between the device and HCPRINT, thus speeding printing.

HCPRINT request initiation logic has been improved, resulting in substantial reductions in resource consumption at sites where devices are routinely left unavailable for printing.

The data streams sent to a device have been optimized to reduce the numbers of characters required to produce the correct print image. Depending on the souce data, this may reduce the number of interactions with the device and hence both transmission and print times.

MULTIPLE LANGUAGE SUPPORT

The system command *)NLT* has been implemented to provide system-defined language tables. This enables programmers whose mother tongue is not English to have APL system and error messages displayed in their own language. Release 19 includes support for English, French, and German.

Entering *)NLT* without an argument, returns the national language table in use. *)NLT* **table** sets it; **table** must be one of

ENGLISH
FRANCAIS
DEUTSCH

ENGLISH is the setting in effect when you sign on.

Examples:

```
      )NLT                    Current NLT setting
IS ENGLISH

      )NLT FRANCAIS           Change NLT to FRANCAIS.
C'ETAIT ENGLISH               Note the reply is in French.
      1 2 + 3 4 5             Error messages are now in French.
ERREUR DE LONGUEUR
      1 2 + 3 4 5
        ^
      ⎕ER                     ⎕ER also reflects NLT setting
5 ERREUR DE LONGUEUR
      1 2 + 3 4 5
        ^
      )NLT DEUTSCH            Change NLT to DEUTSCH
WAR FRANCAIS

      )XYZABC
UNGUELTIGE SYSTEMANWEISUNG

      )NLT ENGLISH
WAS DEUTSCH

      )XYZABC
INCORRECT COMMAND
```

AUXILIARY PROCESSORS

DB2 Interface (AP127)

AP127 is an auxiliary processor which provides an interface to IBM's relational data base product DB2. The functionality is very close to that of APL2's AP127.

In essence, AP127 provides the ability to incorporate SQL statements into an APL program, thus adding SHARP APL to the list of languages that can be used to write SQL/DB2 applications.

This auxiliary processor will form the basis of VIEWPOINT's interface to DB2 files.

TSIO

A dataset created by an ordinary TSIO user has a name which contains an encoding of the user's number. The *DSN* function in workspace 1854339 *TSIO* decodes this to produce the user's number in order to return the name of the dataset. In releases prior to Release 19, the function did this for all datasets having the TSIO prefix. Now it does not do this for names which are not valid encodings.

DSN is called from function *DATASETS*. This change will be noticeable in the output produced by *DATASETS* in that no attempt will be made to decode some names. For more information, see the SHARP APL/370 TSIO User's Guide (publication code: 0461 8703 E19).

AVAM

Extensions are provided to expand the facilities of AVAM; in particular, to allow the use of AVAM as a PLU with multiple sessions.

The new command *OPENACBT* (Open ACB Transparent) causes AVAM to avoid setting either the CID or USR fields in either the RPL or NIB structures, for any session(s) conducted thrugh the ACB thus opened. This enables users to conduct multiple sessions from the same ACB.

The *EXECRPL* command has been extended to allow a wider range of RPL-based operations to be performed. The previous syntax of the command allowed either:

HDR RPL DATA or *HDR RPL NIB*

This is now changed to:

HDR RPL [*NIB*[*BIND*]] [*DATA*] [*ADAT*]

The [*NIB*[*BIND*]], [*DATA*], and [*ADAT*] fields may be specified regardless of the presence of each other, although VTAM requirements will control which of these fields must be present for a particular VTAM request. The *BIND* field may be specified when the *NIB* field is present, thus enabling bind parameters to be negotiated.

Support for the *LOSTERM* exit routine is provided. An RPL image is returned which has the *LOSTERM* return code placed in the RPL RPLFDBK2 field.

For all exit routines, including the new *LOSTERM* routine, AVAM now places the CID and USERFLD values from the exit parameter list in the RPL RPLARG and RPLUSFLD fields respectively to identify the session to which the exit relates.

VSDI

Workspace 1 *VSDI*

There is a new user workspace 1 *VSDI* which contains single functions for using VSDI. Previously, the functions were available only in the maintenance workspace 1854339 *VSAM*.

Initial POINT/READ not required

Previously, it was not possible to perform a WRITE operation as the first operation after opening a non-empty file. An initial POINT or READ operation was required. This anomaly has been removed.

RBA Access

Record Byte Address (RBA) can now be used with the VSDI *PO*, *R*, and *RU* commands to access VSAM records.

For example, `'PO:',⎕AV[⎕IO+256 256 256 256⊤RBA]` positions VSAM so that the next read receives the record that is at the RBA specified.

FCAP

FCAP now supports 31-bit subroutines, that is, subroutines written to run above the line or to reference data stored above the line.

IDSH

IDSH now supports all levels of VTAM.

Change to Command Line

The session manager command line (first row of the standard screen) has been changed slightly. The status field (i.e., the first protected field on the line) has been moved one character to the left.

The leftmost position of the command area (the second field on the line) is also moved one character to the left, and has thus been lengthened by one character. The other fields on the line are unchanged.

This change may affect certain applications, such as intelligent terminal interfaces, that depend on an understanding of the exact from of the datastream that will be sent.

Clipped Mode

When displaying the standard screen in 'clipped' mode (i.e., not 'wrapped') any lines of the session log which do not fit on the one row of the screen will be displayed with a "|" character in the last position. This is done simply to provide a visual indication that theline has been truncated.

New Commands

- Two new session manager commands have been added to permit recall and re-execution of the previous command entered in the command area.

  AG - will recall the last command entered
  RE - will re-execute the last command entered

  These commands can be used anywhere that other commands can be used (for example, in PF key definitions).

- A new command *CURS* has been added to permit some control of the output cursor.

Extended Commands

The *ENTRY* command has been extended so that it can be applied to force or block entry of the command line.

The *INS* command has been extended, primarily to make it possible to amend the command area field.

New Command Line

A new type of line is now formally maintained (in addition to 'session log line' and 'input line'): 'command line'. This type of line represents the contents of the standard screen command area. Two command lines are maintained:

- the currently entered command, referred to by *MI*
- the previously entered command, referred to by *M*

The expressions *M* and *MI* can be used in the *INS* and *CURS* commands.

Full Screen Manager (AP124)

The public workspaces for full screen support have been re-organized, so that the three workspaces 5 *IBM*3270, 5 *AP*124, and 5 *AP*124*E* have been incorporated into two:

5 *AP*124

Contains functions with the same names as those supplied with IBM's *AP*124*X*, functions to control color and highlighting, as well as extended capabilities special to SHARP APL's AP124. (Formerly workspaces 5 *AP*124 and 5 *IBM*3270.)

5 *AP124XT*

The extended version of AP124. The functions in this workspace are like those in 5 *AP124*, except that they use the command chaining feature which allows chaining together of commands and data using the *CTL* variable and no *DAT* variable. There are also some extra functions to allow sending and receiving data as a single enclosed array. (Formerly workspace 5 *AP124E*.)

S-Task Interface (AP1)

An attention message □*AV*[□*IO*+0 0 2 0] sent to AP1 before the S-task is signed on now results in a normal acknowledgement □*AV*[□*IO*+0 6 0 0 0 0]. Previously the response was □*AV*[□*IO*+0 6 0 0 3 0] (S-task not signed on to APL).

□*BOUNCE* for S-tasks:

In releases prior to Release 19, a task which had output waiting for transmission, would not be bounced from the system until the output had been accepted by a recipient. This resulted in the possibility of a task which would never terminate. In Release 19, a second □*BOUNCE* of an S-task will discard the pending output, and terminate the task.

EPH -- Support for Asynchronous Terminals

EPH (Emulation Program Handler) provides support for asynchronous terminals in a module separate from the APL interpreter and scheduler. This modularity makes the whole system more reliable because a failure in asynchronous support will no longer cause failure of APL. It further provides flexibility for a site who may now choose to bring EPH up and down separately from APL, or to run multiple EPH's, or none.

This provides the groundwork for possible future enhancements for specialized terminals; for instance, intelligent terminals or microcomputers. It will also enable I.P. Sharp Associates to make performance enhancements in the interpreter in future releases that are not possible with the current architecture of MPXH.

In Release 19, the use of EPH is optional. Thus, sites may choose when to phase-in use of EPH, in preparation for when MPXH will be desupported in the next release.

The following changes apply for terminal access through EPH. For more information, see the SHARP APL/370 Emulator Program Handler (EPH) User's Guide.

□*RUNS*

The result of □*RUNS*[;□*IO*+2] has changed from being the user number of the associated APL task to the integer 9 (i.e. the task is initiated by AP9, EPH).

⍞ *INPUT*

The rule for applying input editing to the result of a reference to ⍞ has changed. This change only affects applications which issue prompts by the method:

```
⍞←PROMPT ⋄ RSP←⍞
```

in which the prompt contains a character such as BACKSPACE, LINEFEED or some undisplayable character. The rule change does not affect applications that use *⎕ARBOUT* '' to force display of the prompt before referencing ⍞.

Previously, the rule was as follows:

- input editing was not applied to the prompt
- input editing was applied to the typist's response
- the first was catenated to the front of the second

For example:

```
      ⍞←'PROMPT:',LF ⋄ ⍴⎕←⍞
PROMPT:
      RESPONSE
PROMPT:
      RESPONSE
16
```

In Release 19, the new rule is as follows:

- the prompt is catenated to the front of the typist's response
- input editing is applied to the result of this catenation

The same example now reads:

```
      ⍞←'PROMPT:',LF ⋄ ⍴⎕←⍞
PROMPT:
      RESPONSE
PROMPT:RESPONSE
15
```

Note that this rule has been in effect for some years for terminals other than those connected through IPSANET (e.g., 3270 terminals).

⎕HT

One instance of reassignment of *⎕HT* is not honoured, which is *)COPY* of *⎕HT* from a saved workspace. The change to *⎕HT* is not "noticed", and so the system's tab settings are not altered. This can be remedied by executing *⎕HT←⎕HT* after the *)COPY*.

MISCELLANEOUS

ISO Conformance

A number of changes have been made to SHARP APL to bring it into conformance with the ISO APL standard, ISO/DIS8485.

In *F/* for the functions + - × ÷ ⌊ ⌈, elements are processed from right to left, in conformance with the ISO APL standard.

↑ and ↓ now accept right arguments of rank up to 63.

Improved CPU Accounting and Timing Precision

Timer services have been redesigned. A consequence is that CPU accounting is considerably more precise: microunit resolution is kept internally, and milliunit resolution is reported in ⎕*AV*[1 5+⎕*IO*]. (The previous resolution was tricentiunits). Thus, fewer CPU cycles are unaccounted for by APL's CPU accounting. Many APL expressions (especially those involving file operations) will appear to cost more in terms of ⎕*AI*[1 5+⎕*IO*], though not in terms of machine resources actually consumed.

Likewise, ⎕*TS* now has millisecond resolution.

Improvements to Mathematical Functions

The precision of the complex floor function has been improved for an argument having the fractional parts of the real and imaginary parts adding to 1 mathematically, but to less than 1 in machine terms. In such cases, the result was less by a unit in either the real or the imaginary part from what it should have been. For example, ⌊¯2.88*J*16.88 now gives ¯3*J*17, previously it gave ¯3*J*16.

The elementary functions are more accurate and faster. The affected primitives are α○ω, α*ω, *ω, ω*0.5, α⊛ω, ⊛ω, α!ω and !ω. Included in α○ω are all the subfunctions: sine, cosine, inverse hyperbolic tangent, etc.

For a detailed description of improvements to a particular function in this class, see E.E.McDonnell's paper "A Perfect Square Root Routine" in the APL86 Conference Preceedings, pp. 289-294.

Minor Changes and Corrections

Various anomalies related to function definition and code string building have been corrected.

System command verbs are accepted in either of the first two alphabets. For example,)*LIB*,)*LIB*,)*LIB* all work correctly.

PROFILE Security

PROFILE workspaces are now protected in the same way that *CONTINUE* workspaces are: the workspace cannot be loaded from another user number, unless the sign-on lock for the workspace owner number is specified.

Storage for *PROFILE* workspaces are no longer billed.

(2 ⎕*WS* 4)[⎕*IO*+1], workspace save date, is now zero for unsaved workspaces.

5 ⎕*WS* *ω* now reports class 2 for local shared variables, even if they do not currently have values.

The derived function *F/* is now more strict in checking the validity of its arguments.

Signoff Report Change

The distributed example signoff report displayed at signoff has been widened to avoid truncation of the display when large amounts of CPU resources have been consumed.

File System Interrupts

File system interrupts now occur after a strong interrupt (two breaks). The purpose is to simplify the writing of S-task drivers, and to bring file system behavior into line with the rest of SHARP APL.

Minor Changes in the ∇ Editor

The caret which points to errors on line 0 now points to the first error on the line, scanning from left to right, rather than simply pointing to the end of the line.

The maximum line number has been raised to 99999.999 from 32699.99.

Typing [⎕n] with n greater than the last line in the function prompts with the appropriate line number following the last line of the function.

For)*MSG*[*N*] and)*OPR*[*N*], all leading blanks but one are suppressed from the transmitted message text.

)*CPULIM* now rejects negative arguments.

Communications

Delay after Failed Sign-on Attempt

After 4 failed sign-on attempts, a 5 second delay is imposed. Increasing delays are imposed until after 10 failures the delay is 15 seconds.

Typing)*OFF* in the not-signed-on state no longer causes a 10 second delay.

)PORT Node Numbers

The *)PORT* system command now treats numbers up to 999 as node numbers. Numbers above 999 are still treated as user numbers. For example, *)PORT* 666 lists all users signed on to node 666, whereas formerly it listed all users signed on to user number 666.

⎕ARBOUT/⎕ARBIN

The integer domain of *⎕ARBOUT* and *⎕ARBIN* are now the integers from 0 to 255. Previously, values in the range 256-511 were also permitted.

⎕ARBOUT/⎕ARBIN for NTOH access

For asynchronous terminal access through the Network Terminal Option Handler (NTOH), the domain of *⎕ARBOUT* and *⎕ARBIN* has been extended from ⍳128 to ⍳256 in the following way: *⎕ARBOUT* 128+*X* has the same effect as *⎕ARBOUT X*.

Obsolescent and Obsolete Facilities

Keywords (*ΔFI ΔVI ΔFMT FE ΔWS ΔFD*) in saved workspaces cause these workspaces to be marked as "damaged" by the workspace utility programs *WSUT* and *WSTC*. Future releases of SHARP APL will not include support for keywords. They should be replaced by their equivalent system functions, and the keywords expunged from the workspace.

Indirect pointers are no longer supported. These are present only in very old saved workspaces (saved before installation of SHARP APL Release 15, circa May 1983) with nonempty *SI* stacks. Such workspaces will be marked as damaged by *WSUT* and *WSTC*. Clearing the *SI* stack via *)SIC*, then resaving the workspace, will repair the damage.

Both of the above support changes are required as part of a planned project to improve interpreter performance.

Several obsolete *⎕OUT* control messages are no longer supported. They are:

```
NOT IN FN DEFN
CHAR ERROR
OPEN QUOTE
```

APPENDIX A: Changes to Internal Form of APL Sentences

APL sentences (lines of APL) are converted to an internal form before they are executed. This conversion process has been redesigned. The following changes are visible to an APL programmer:

1. In releases of SHARP APL prior to Release 19, the internal form (data type) of a numeric literal depended on how it was entered. For example, `1J0` would be represented as a complex constant, even though it displays in the same way internally as `1` or `1.0`.

   In Release 19, numeric literals are represented internally in the form which minimizes the storage for that literal. For example,

   ```
   1
   1.0
   1J0
   ```

   will all be represented internally as boolean.

   Note also that use of such constants to initialize files:

   ```
   (1000⍴1J0) ⎕APPENDR TN
   ```

   no longer results in the desired datatype being created. Constants of unambiguous type such as `1J1` should be used in such applications.

2. Execution of sentences that cannot be parsed will result in a *'SYNTAX ERROR'* similar to that signalled for *'UNMATCHED QUOTES'*. Errors in this class include *UNMATCHED QUOTES*, *UNKNOWN ⎕ NAMES*, third alphabet and national use characters found outside comments or literals, and badly formed numeric literals such as `1..9E¯2.2`.

2. Syntax errors caused by errors described in (3) preserve the line label. In releases of SHARP APL prior to Release 19, labelled lines which contained unbalanced quotes lost their label and a more global value of the label was used. This could result in erroneous function execution.

4. *DEFN ERROR* is now reported if a function has more than 2047 labels, or 32767 lines.

5. Expressions such as `1JULY 85` and `2EASY 2` are treated as badly formed numeric literals as in (3). In releases of SHARP APL prior to Release 19, the leading numeric portion (1 and 2) was treated as a left argument to the function (*JULY*, *EASY*).

With 19 elements are processed from right to left, in conformance with the ISO APL standard. Mathematically, the results should be the same, but in practice, there may be differences introduced by the limitations of computer arithmetic.

Examples:

1. Different results due to accumulation of rounding error. For example:

```
+/5 1E20 ¯1E20 ¯5
```

   This example gives the value ¯5 with Release 17 and the value 5 with Release 19. Note: the result of reduction may differ from the value of the last element in the result of a scan for this reason.

2. Different results due to non-representable intermediate values. For example:

```
×/0 1E70 1E10
```

   gives the value 0 in Release 17 and signals *DOMAIN ERROR* in Release 19. The opposite is true for ×/1E70 1E10 0.

3. Performance can change depending on data, when the result type changes during computation of the result. For example:

```
DATA←(20⍴10), 10000⍴0 1 2 3
```

   ×/*DATA* is considerably faster in Release 19 compared to Release 17, whereas the opposite is true for ×/⍎*DATA*.